Search Results

Search Results for: **[(free* or deallocat* or de-allocat* or releas*)<AND>((jit and (optimiz* or profil*) and (hot or cold or inactive or frequent* or infrequent*)))]**
Found **161** of **130,565 searched.**

Search within Results

Sort by: Title Publication Publication Date Score Binder

Results 1 - 20 of 161 short listing

Prev Page 1 2 3 4 5 6 7 8 9 Next Page

1 Open runtime platform: flexibility with performance using interfaces 100%
Michal Cierniak , Brian T. Lewis , James M. Stichnoth
Proceedings of the 2002 joint ACM-ISCOPE conference on Java Grande November 2002

According to conventional wisdom, interfaces provide flexibility at the cost of performance. Most high-performance Java virtual machines today tightly integrate their core virtual machines with their just-in-time compilers and garbage collectors to get the best performance. The Open Runtime Platform (ORP) is unusual in that it reconciles high performance with the extensive use of well-defined interfaces between its components. ORP was developed to support experiments in dynamic compilation, garb ...

2 A brief history of just-in-time 100%
John Aycock
ACM Computing Surveys (CSUR) June 2003
Volume 35 Issue 2

Software systems have been using "just-in-time" compilation (JIT) techniques since the 1960s. Broadly, JIT compilation includes any translation performed dynamically, after a program has started execution. We examine the motivation behind JIT compilation and constraints imposed on JIT compilation systems, and present a classification scheme for such systems. This classification emerges as we survey forty years of JIT work, from 1960--2000.

3 Technical correspondence: The simplest heuristics may be the best in Java JIT compilers 100%
Jonathan L. Schilling

ACM SIGPLAN Notices February 2003
Volume 38 Issue 2

The simplest strategy in Java just-in-time (JIT) compilers is to compile each Java method the first time it is called. However, better performance can often be obtained by selectively compiling methods based on heuristics of how often they are likely to be called during the rest of the program's execution. Various heuristics are examined when used as part of the Caldera UNIX Java JIT compiler. The simplest heuristics involving the number of times the method has executed so far and the size of th ...

4 Techniques for obtaining high performance in Java programs 100%

Iffat H. Kazi , Howard H. Chen , Berdenia Stanley , David J. Lilja
ACM Computing Surveys (CSUR) September 2000
Volume 32 Issue 3

This survey describes research directions in techniques to improve the performance of programs written in the Java programming language. The standard technique for Java execution is interpretation, which provides for extensive portability of programs. A Java interpreter dynamically executes Java bytecodes, which comprise the instruction set of the Java Virtual Machine (JVM). Execution time performance of Java programs can be improved through compilation, possibly at the expense of portabili ...

5 Fast, effective code generation in a just-in-time Java compiler 100%

Ali-Reza Adl-Tabatabai , Michał Cierniak , Guei-Yuan Lueh , Vishesh M. Parikh , James M. Stichnoth
ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 1998 conference on Programming language design and implementation May 1998
Volume 33 Issue 5

A "Just-In-Time" (JIT) Java compiler produces native code from Java byte code instructions during program execution. As such, compilation speed is more important in a Java JIT compiler than in a traditional compiler, requiring optimization algorithms to be lightweight and effective. We present the structure of a Java JIT compiler for the Intel Architecture, describe the lightweight implementation of JIT compiler optimizations (e.g., common subexpression elimination, register allocation, and elim ...

6 A small hybrid JIT for embedded systems 100%

Geetha Manjunath , Venkatesh Krishnan
ACM SIGPLAN Notices April 2000
Volume 35 Issue 4

Just-In-Time (JIT) Compilation is a technology used to improve speed of virtual machines that support dynamic loading of applications. It is better known as a technique that accelerates Java programs. Current JIT compilers are either huge in size or compile complete methods of the application requiring large amounts of memory for their functioning. This has made Java Virtual Machines for embedded systems devoid of JIT compilers. This paper explains a simple technique of combining interpretation ...

7 Effectiveness of cross-platform optimizations for a java just-in-time compiler 100%

Kazuaki Ishizaki , Mikio Takeuchi , Kiyokuni Kawachiya , Toshio Suganuma , Osamu Gohda , Tatsushi Inagaki , Akira Koseki , Kazunori Ogata , Motohiro Kawahito , Toshiaki Yasue , Takeshi Ogasawara , Tamiya Onodera , Hideaki Komatsu , Toshio Nakatani
ACM SIGPLAN Notices , Proceedings of the 18th ACM SIGPLAN conference on

Object-oriented programing, systems, languages, and applications October 2003
Volume 38 Issue 11

This paper describes the system overview of our Java Just-In-Time (JIT) compiler, which is the basis for the latest production version of IBM Java JIT compiler that supports a diversity of processor architectures including both 32-bit and 64-bit modes, CISC, RISC, and VLIW architectures. In particular, we focus on the design and evaluation of the cross-platform optimizations that are common across different architectures. We studied the effectiveness of each optimization by selectively disabling ...

8 Practicing JUDO: Java under dynamic optimizations 100%
Michał Cierniak , Guei-Yuan Lueh , James M. Stichnoth
ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation May 2000
Volume 35 Issue 5

A high-performance implementation of a Java Virtual Machine (JVM) consists of efficient implementation of Just-In-Time (JIT) compilation, exception handling, synchronization mechanism, and garbage collection (GC). These components are tightly coupled to achieve high performance. In this paper, we present some static anddynamic techniques implemented in the JIT compilation and exception handling of the Microprocessor Research Lab Virtual Machine (MRL VM), ...

9 Java annotation-aware just-in-time (AJIT) complilation system 99%
Ana Azevedo , Alex Nicolau , Joe Hummel
Proceedings of the ACM 1999 conference on Java Grande June 1999

10 Split-stream dictionary program compression 99%
Steven Lucco
ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2000 conference on Programming language design and implementation May 2000
Volume 35 Issue 5

This paper describes split-stream dictionary (SSD) compression, a new technique for transforming programs into a compact, interpretable form. We define a compressed program as interpretable when it can be decompressed at basic-block granularity with reasonable efficiency. The granularity requirement enables interpreters or just-in-time (JIT) translators to decompress basic blocks incrementally during program execution. Our previous approach to interpretable compression, the Byte-coded RISC ...

11 Design, implementation, and evaluation of optimizations in a just-in-time compiler 99%
Kazuaki Ishizaki , Motohiro Kawahito , Toshiaki Yasue , Mikio Takeuchi , Takeshi Ogasawara , Toshio Suganuma , Tamiya Onodera , Hideaki Komatsu , Toshio Nakatani
Proceedings of the ACM 1999 conference on Java Grande June 1999

12 Efficient Java exception handling in just-in-time compilation 99%
SeungIl Lee , Byung-Sun Yang , Suhyun Kim , Seongbae Park , Soo-Mook Moon , Kemal Ebcioğlu , Erik Altman
Proceedings of the ACM 2000 conference on Java Grande June 2000

13 Memory system behavior of Java programs: methodology and analysis 98%

Jin-Soo Kim , Yarsun Hsu
ACM SIGMETRICS Performance Evaluation Review , Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems June 2000
Volume 28 Issue 1

This paper studies the memory system behavior of Java programs by analyzing memory reference traces of several SPECjvm98 applications running with a Just-In-Time (JIT) compiler. Trace information is collected by an exception-based tracing tool called JTRACE, without any instrumentation to the Java programs or the JIT compiler.First, we find that the overall cache miss ratio is increased due to garbage collection, which suffers from higher cache misses compared to the application. ...

14 Stride prefetching by dynamically inspecting objects 97%
Tatsushi Inagaki , Tamiya Onodera , Hideaki Komatsu , Toshio Nakatani
ACM SIGPLAN Notices , Proceedings of the ACM SIGPLAN 2003 conference on Programming language design and implementation May 2003
Volume 38 Issue 5

Software prefetching is a promising technique to hide cache miss latencies, but it remains challenging to effectively prefetch pointer-based data structures because obtaining the memory address to be prefetched requires pointer dereferences. The recently proposed stride prefetching overcomes this problem, but it only exploits *inter-iteration* stride patterns and relies on an off-line profiling method.We propose a new algorithm for stride prefetching which is intended for use in a dynamic ...

15 Efficient Java RMI for parallel programming 97%
ACM Transactions on Programming Languages and Systems (TOPLAS) November 2001
Volume 23 Issue 6

Java offers interesting opportunities for parallel computing. In particular, Java Remote Method Invocation (RMI) provides a flexible kind of remote procedure call (RPC) that supports polymorphism. Sun's RMI implementation achieves this kind of flexibility at the cost of a major runtime overhead. The goal of this article is to show that RMI can be implemented efficiently, while still supporting polymorphism and allowing interoperability with Java Virtual Machines (JVMs). We study a new approach f ...

16 HBench:Java: an application-specific benchmarking framework for Java virtual machines 97%
Xiaolan Zhang , Margo Seltzer
Proceedings of the ACM 2000 conference on Java Grande June 2000

17 Code scheduling: Integrated prepass scheduling for a Java Just-In-Time compiler on the IA-64 architecture 96%
Tatsushi Inagaki , Hideaki Komatsu , Toshio Nakatani
Proceedings of the international symposium on Code generation and optimization: feedback-directed and runtime optimization March 2003

We present a new integrated prepass scheduling (IPS) algorithm for a Java Just-In-Time (JIT) compiler, which integrates register minimization into list scheduling. We use backtracking in the list scheduling when we have used up all the available registers. To reduce the overhead of backtracking, we incrementally maintain a set of candidate instructions for undoing scheduling. To maximize the ILP after undoing scheduling, we select an instruction chain with the smallest increase in the total exec ...

18 Reducing virtual call overheads in a Java VM just-in-time compiler 96%

Junpyo Lee , Byung-Sun Yang , Suhyun Kim , Kemal Ebcioğlu , Erik Altman , Seungil Lee , Yoo C. Chung , Heungbok Lee , Je Hyung Lee , Soo-Mook Moon

ACM SIGARCH Computer Architecture News March 2000
Volume 28 Issue 1

Java, an object-oriented language, uses *virtual methods* to support the extension and reuse of classes. Unfortunately, virtual method calls affect performance and thus require an efficient implementation, especially when just-in-time (JIT) compilation is done. *Inline caches* and *type feedback* are solutions used by compilers for dynamically-typed object-oriented languages such as SELF [1, 2, 3], where virtual call overheads are much more critical to performance than in Java. Wi ...

19 Dynamic metrics for java 96%

Bruno Dufour , Karel Driesen , Laurie Hendren , Clark Verbrugge

ACM SIGPLAN Notices , Proceedings of the 18th ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications October 2003
Volume 38 Issue 11

In order to perform meaningful experiments in optimizing compilation and run-time system design, researchers usually rely on a suite of benchmark programs of interest to the optimization technique under consideration. Programs are described as *numeric*, *memory-intensive*, *concurrent*, or *object-oriented*, based on a qualitative appraisal, in some cases with little justification. We believe it is beneficial to quantify the behaviour of programs with a concise and precisely ...

20 A dynamic optimization framework for a Java just-in-time compiler 95%

Toshio Suganuma , Toshiaki Yasue , Motohiro Kawahito , Hideaki Komatsu , Toshio Nakatani

ACM SIGPLAN Notices , Proceedings of the 16th ACM SIGPLAN conference on Object oriented programming, systems, languages, and applications October 2001
Volume 36 Issue 11

The high performance implementation of Java Virtual Machines (JVM) and just-in-time (JIT) compilers is directed toward adaptive compilation optimizations on the basis of online runtime profile information. This paper describes the design and implementation of a dynamic optimization framework in a production-level Java JIT compiler. Our approach is to employ a mixed mode interpreter and a three level optimizing compiler, supporting quick, full, and special optimization, each of which has a differ ...

Results 1 - 20 of 161 short listing

Prev Page 1 2 3 4 5 6 7 8 9 Next Page